

# Analyzing Dshield Logs Using Fully Automatic Cross-Associations

Anh Le<sup>1</sup>

<sup>1</sup>Donald Bren School of Information and Computer Sciences  
University of California, Irvine  
Irvine, CA, 92697, USA  
*anh.le@uci.edu*

## 1 Introduction

Filtering malicious traffic based on communication sources [4, 5] is one of the approaches to protect network infrastructure from attacks on the Internet. The key challenge of this approach is the accuracy of the source-based filters (or blacklists). Inaccurate filters not only result in poor defense against malicious traffic but also affect benign traffic. In particular, the malicious traffic that get through might still cause severe damage to the network infrastructure while users whose benign traffic blocked by such filters of a network can not use services offered by the network.

In this work, we aim to improve the accuracy of filtering by analyzing and exploiting characteristics of malicious traffic. Our work makes use of the fully automatic cross-associations (CA) [2] algorithm – a clustering algorithm that finds row and column groups of sparse binary matrices simultaneously and in a fully automatic manner. In particular, we apply the CA algorithm on a Dshield dataset [3] containing millions of security log entries contributed by more than a thousand of contributors across the Internet. From the result of the algorithm, we identify malicious communication sources and contributors that are highly correlated to each other. We then use the uncovered correlated communication sources and contributors for the following two purposes:

1. Botnet detection: Botnets are infected hosts on the Internet that are controlled by attackers to carry out a wide range of attacks, such as scanning, worm propagations, and distributed denial of service (DDoS) attacks. In fact, large-scale attacks on the Internet, such as DDoS attacks, usually require correspondingly large botnets. Consequently, successful identifications of botnets significantly help to block such large-scale attacks accurately and in a timely manner. Our botnet detection is by examining the correlated communication sources.
2. Attack prediction: An attack mounted towards a specific contributor at a particular point in time might be mounted towards another contributor, who is correlated to the former contributor, at a later time. This is because correlated contributors observe similar attack sources; thus, they are often targets of similar attacks. We predict attacks towards a particular contributor by examining attack sources observed by this contributor and the contributors that are correlated to it.

Our preliminary results show that the application of the CA algorithm on Dshield dataset consistently finds clusters of communication sources (both source IP addresses and subnets) and

contributors. More importantly, there exists clusters with very high densities, meaning that sources and contributors belonging to these clusters are highly correlated. Furthermore, by analyzing these dense clusters over time, we successfully identify strong candidates of botnets. These are source subnets that belong to dense clusters and persistently attack multiple contributors over time.

The rest of this study is organized as follows: Section 2 briefly describes the CA algorithm and the Dshield dataset that we have; we also explain the three different levels of granularity of experiments that we conducted. Section 3 reports preliminary results of this study. Finally, Section 4 gives our concluding remarks and points out steps that should be taken in a subsequent study.

## 2 Background and Experiments

### 2.1 Fully Automatic Cross-Associations Algorithm

Proposed by Chakrabarti *et al.* [2], the CA algorithm is an off-line clustering algorithm that operates on a sparse binary matrix. It decomposes an input sparse binary matrix into disjoint row and column clusters such that the rectangular intersections of these clusters are homogeneous. One significant property of the CA algorithm is that it is *parameter-free*. In other words, this algorithm does not require users to set any parameters, and its only input is the binary matrix; hence, it is fully automatic in this sense. Furthermore, the algorithm *scales linearly* with the input matrix size. This is particularly important to our work since we deal with matrices of sizes up to a million by a million. In our work, we currently consider the CA algorithm as a *black box* that helps us to find row and column clusters.

### 2.2 Dshield Dataset

Dshield [3] is a large-scale security log sharing system. It has more than 1700 contributors who provide a daily stream of 30 million security log entries. These log entries come from a variety of devices, such as firewalls and intrusion detection systems. Each entry is a tuple consisting of the following fields: date, time, contributor identification number (*c\_id*), count, source IP address (*src\_ip*) and port number, destination IP address (*dst\_ip*) and port number, protocol number, and flags. In addition to these fields, for each entry in our log, we made queries to the ARIN WHOIS database [1] to acquire its source and destination subnets (*src\_subnet* and *dst\_subnet*), ASes, and countries. We also pre-processed our log to exclude noises and erroneous data that may exist, such as entries belonging to invalid, non-routable, or unassigned IP address spaces and entries whose *c\_id*'s indicate test or anonymous submissions. Finally, we have 6 months of Dshield log from May to October 2008; however, in this study, we only examine the last 12 days of log of October 2008: from the 20<sup>th</sup> to the 31<sup>st</sup>.

### 2.3 Experiments

We conduct three sets of experiments. For each set of the experiments, we use matrices of a different type – corresponding to a different level of granularity – as inputs to the CA algorithm. For the first set of the experiments, we extract the pairs (*dst\_ip*, *src\_ip*) from the log entries of each day, so the inputs to the CA algorithm in this case are 12 matrices (of the 12 days) whose rows and columns are destination and source IP addresses, respectively. Similarly, the second and the third sets of the experiments involve the pairs (*c\_id*, *src\_ip*) and (*c\_id*, *src\_subnet*); hence, the inputs to the CA algorithm in these two cases are two sets of 12 matrices: the first set has rows and columns of the matrices corresponding to contributor ID numbers and source IP addresses while the second

set has rows and columns of the matrices corresponding to contributor ID numbers and source subnets.

The typical sizes of matrices containing pairs of  $(dst\_ip, src\_ip)$ ,  $(c\_id, src\_ip)$ , and  $(c\_id, src\_subnet)$  of log entries of 1 day are  $(2 \cdot 10^5 \times 8 \cdot 10^5)$ ,  $(5 \cdot 10^2 \times 8 \cdot 10^5)$ , and  $(5 \cdot 10^2 \times 6 \cdot 10^3)$ , respectively. The CA algorithm is implemented in Matlab. The experiments are run on a dual-CPU machine –  $2 \times 2.8$  Ghz Intel Quad Core Xeon – that has 32 GBs of RAM. For a matrix containing pairs of  $(dst\_ip, src\_ip)$  or  $(c\_id, src\_ip)$  of log entries of 1 day, the CA algorithm takes about 1 hour to finish. Meanwhile, for a matrix containing pairs of  $(c\_id, src\_subnet)$  of log entries of 1 day, the CA algorithm only takes about 15 minutes to finish. We also observe that during the clustering process, the CA algorithm only utilizes 1 core of the CPU.

It is important to note that 4 one-to-one mappings that map distinct  $dst\_ip$ 's,  $c\_id$ 's,  $src\_ip$ 's, and  $src\_subnet$ 's to distinct small integers are required as a preprocessing step that produces compatible input matrices for the CA algorithm. Furthermore, the CA algorithm does not take into account duplicated entries. This means that a pair of  $(dst\_ip, src\_ip)$ , for instance, that appears multiple times during a day is treated the same as it appears only once by the CA algorithm.

### 3 Preliminary Results

#### 3.1 Clustering

Figure 1 shows the results of running the CA algorithm on 3 matrices which contain pairs of  $(dst\_ip, src\_ip)$ ,  $(c\_id, src\_ip)$ , and  $(c\_id, src\_subnet)$  of 1-day log entries (we show the result of October 20<sup>th</sup>; nevertheless, the results of the other days are similar). Overall, the algorithm successfully identify clusters of rows and columns for all 3 inputs. The numbers of row and column clusters for all 3 matrices vary between 5 and 20 over the 12 days.

Individually, for the  $(c\_id, src\_ip)$  and  $(c\_id, src\_subnet)$  matrices, the resulted clusters are more visible than the clusters of the  $(dst\_ip, src\_ip)$  matrix. This implies either that the  $(dst\_ip, src\_ip)$  matrix naturally does not have clear clusters comparing to the other two matrices, or that the CA algorithm may not work well on this matrix.

We also observe that all the clusters founded by the algorithm are packed at the lower right corner; meanwhile, there are usually empty clusters that cover a large amount of space on the upper left corner. The large size of the empty space is as expected since the inputs are sparse matrices; nonetheless, this does indicate that the majority of communication sources and destinations has very little correlation.

We further examine the clusters by calculating the density of each cluster identified by the CA algorithm in all 3 cases. A density of a cluster equals to the result of dividing the occupied space of the clusters – the black area – and the area of the cluster. Figure 2 plots the histograms and the CDFs of the densities of the clusters of all 3 matrices.

From the histograms of Figures 2(a), 2(c), and 2(e), it can be observed that the clusters of the  $(dst\_ip, src\_ip)$  matrix generally have much lower densities comparing to the clusters of the other two matrices. In particular, the CDF plots from Figures 2(b), 2(d), and 2(f) indicate that 100% of the clusters of the  $(dst\_ip, src\_ip)$  matrix have densities less than 50%; meanwhile, 10% of the clusters of the  $(c\_id, src\_ip)$  matrix and 30% of the clusters of the  $(c\_id, src\_subnet)$  matrix have densities higher than 50%.

It is important to note that highly dense clusters are more valuable to us as they indicate that their corresponding rows and columns are highly correlated. For example, a highly dense  $(c\_id, src\_subnet)$  cluster indicates that its contributors are highly correlated because they observe

similar attack source subnets; likewise, its observed source subnets are highly correlated as well because they attack the same set of contributors.

In summary, the CA algorithm produces the most visible and dense clusters when running on the  $(c\_id, src\_subnet)$  matrices. It produces the least visible and dense clusters when running on the  $(dst\_ip, src\_subnet)$  matrices. Overall, it manages to identify a significant numbers of both row and column clusters varying from 5 to 20 in all three cases.

### 3.2 Botnet Detection

In order to identify candidates of botnets, we choose to analyze the clusters of the  $(c\_id, src\_subnet)$  matrices (out of the three types of matrices) over time. There are 3 main reasons for this:

- The CA algorithm produces the most dense clusters when running on the  $(c\_id, src\_subnet)$  matrices; as a result, it implies the highest correlation of the attack sources.
- The clusters of the  $(dst\_ip, src\_ip)$  matrices have low densities, as indicated by Figure 2(a), as a result, giving weak correlation.
- It first seems reasonable to carry out the botnet detection analysis on the clusters of the  $(c\_id, src\_ip)$  matrices because these matrices have several highly dense clusters, as indicated by Figure 2(c). However, according to our previous study of the Dshield log, we found that  $src\_ip$ 's rarely occur twice over time due to IP dynamic allocations. This discourages any analysis of individual IPs over time.

For the clusters of the  $(c\_id, src\_subnet)$  matrices, we focus on the ones with densities at least 75%. This threshold gives us a nice property that on average, any pair of source subnets of the same cluster has 50% of overlap in terms of the number of contributors that they attack. This is because on average, each source subnet attacks 75% of the contributors of its cluster. Nevertheless, one can use any other reasonable (high) threshold.

Furthermore, we only analyze clusters which have more than one contributor because we consider the ones having only one contributor as degenerate cases. In fact, for each day of the 12-day period, there are usually clusters which have only 1 contributor but more than 80% of the source subnets; hence, taking them into account is equivalent to considering most of the available subnets.

On average, there are 2 clusters whose densities are at least 75% and having more than 1 contributor per day. We track the appearance of the source subnets of these clusters over the course of 12 days. Figure 3(a) and 3(b) plot the histogram and the CDF of the number of appearances of these clusters. It can be observed from Figure 3(a) that the majority of the source subnets appears more than once, and there are source subnets that appear in most of the 12 days. More specifically, Figure 3(b) indicates that 70% of the source subnets appear in more than one day; about 30% of the source subnets ( $\sim 400$  subnets) appear in more than 3 days; and interestingly, about 5% of the source subnets ( $\sim 80$  subnets) appear in more than 90% of the 12-day period ( $\sim 10$  days). We consider these last source subnets as strong candidates of botnets.

In summary, tracking the appearance of the attack source subnets of clusters whose densities are at least 75% reveals that the majority of these subnets appears in more than 1 day. More importantly, there is a significant number of source subnets which appears in more than 90% of the 12-day period. These source subnets are strong candidates of botnets because they persistently attack similar groups of contributors; however, further analyses are needed to conclude whether they are botnets.

### 3.3 Duplicated Entries

As previously mentioned in Section 2.3, the CA algorithm does not take into account duplicated log entries. However, the duplication should be considered because it may affect both the botnet detection and the attack prediction. One important reason is for this that pairs appearing more than once in a log indicate more active and aggressive attackers.

One of the ways that address this issue is to incorporate the numbers of duplication (or weights) of the pairs into the CA algorithm. In this study, we examine a different way, which pre-process the pairs based on their weights. We study different thresholds of the weight to selectively construct the input matrices. In particular, for a weight threshold, we only select pairs whose weights are above the threshold to construct the matrices.

Figure 4 plots the histograms of the weights of all three different pairs ( $dst\_ip, src\_ip$ ), ( $c\_id, src\_ip$ ), and ( $c\_id, src\_subnet$ ) of 1-day log entries. The histograms show that the majority of the pairs has weight 1, *i.e.* they appear only once during the day. More specifically, the collected statistics show that on average, 75% of the ( $dst\_ip, src\_ip$ ) pairs have weight 1 out of all ( $dst\_ip, src\_ip$ ) pairs; similarly, 63% of the ( $c\_id, src\_ip$ ) pairs and 40% of the ( $c\_id, src\_subnet$ ) pairs have weight 1 out of all of their respective pairs. This indicates that just by filtering out the pairs with weight 1, the results of the analyses may differ significantly.

We filtered out pairs with weight 1 and ran the CA algorithm on the matrices constructed using the remaining pairs. Figure 5 plots side-by-side the results of running the CA algorithm on matrices constructed by all pairs and by pairs appearing more than once. Our initial observation is that running the algorithm on the latter results in a fewer number of clusters. Further analyses are needed to examine the true impact of this filtering.

## 4 Conclusion and Future Work

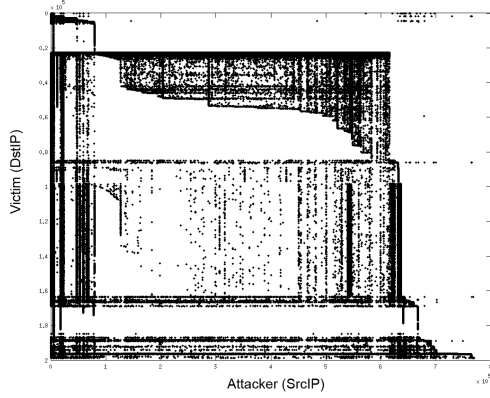
In this study, we verified that the CA algorithm can successfully identify clusters of both communication sources (source IP addresses and subnets) and destinations (destination IP addresses and contributors) of Dshield log entries. More importantly, for input matrices of ( $c\_id, src\_subnet$ ) pairs, there exist clusters with very high densities. This in turn indicates high correlation of both the contributors and the attack source subnets. In addition, we found strong candidates of botnets by tracking the attack source subnets of highly dense clusters. Specifically, the candidates of botnets are the source subnets that we found persistently attacking the same groups of contributors over time. Finally, we pointed out the importance of the high numbers of appearances of log entries. Our initial investigation indicates that filtering out entries which appear only once during a day may significantly impact the later analyses.

We end this study by describing some important future work:

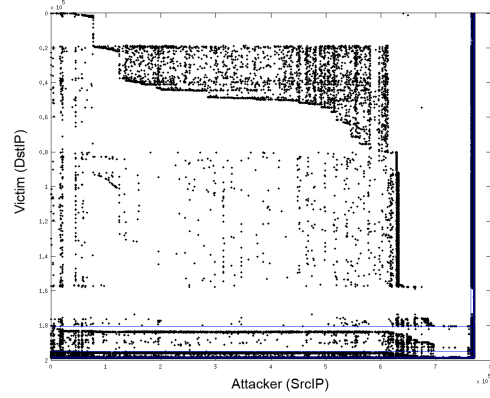
- Verify if the uncovered candidates of botnets are really botnets. This can be done by tracking back the corresponding source IP addresses and the destination port numbers based on the uncovered source subnets and then examining the types of the attacks.
- Analyze the correlation of the contributors to make attack prediction. Initially, the prediction should include source subnets that are botnet candidates because they are consistently observed by the correlated contributors.
- Apply the same analyses but on the entries that appear more than once to understand the true impact of weights. The results of the analyses done here may give more accurate botnet prediction because they focus on more active and aggressive attackers.

## References

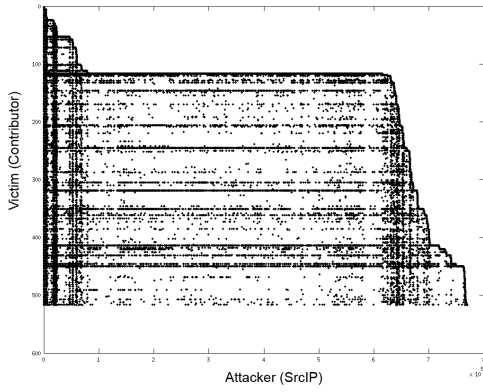
- [1] ARIN WHOIS database. <http://ws.arin.net/whois/>.
- [2] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S. Modha, and Christos Faloutsos. Fully automatic cross-associations. In *KDD '04: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 79–88, Seattle, WA, USA, 2004. ACM.
- [3] Dshield dataset. <http://www.dshield.org/>.
- [4] Fabio Solder, Athina Markopoulou, and Katerina Argyraki. Optimal filtering of source address prefixes: Models and algorithms. In *INFOCOM '09: Proceedings of the 28th IEEE Conference on Computer Communications (to appear)*, Rio de Janeiro, Brazil, April 2009. IEEE.
- [5] Fabio Soldo, Karim El Defrawy, Athina Markopoulou, Balachander Krishnamurthy, and Kobus van der Merwe. Filtering sources of unwanted traffic. In *ITA Workshop '08*, San Diego, CA, USA, January 2008.



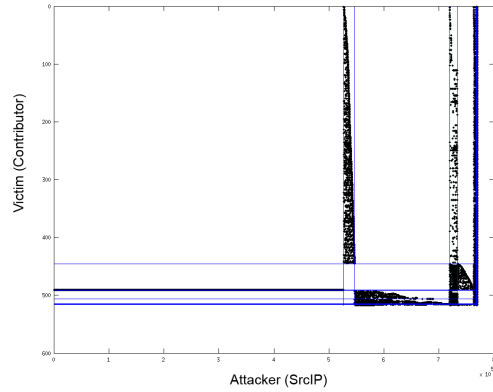
(a) Original  $(dst\_ip, src\_ip)$  Matrix



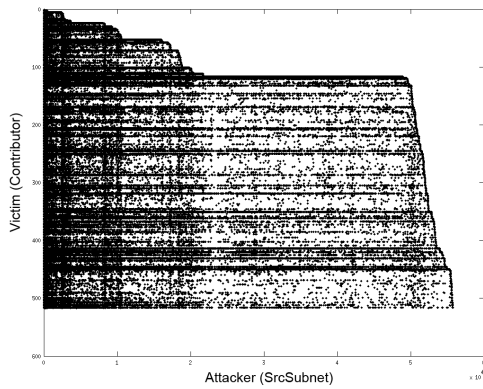
(b) Clustered  $(dst\_ip, src\_ip)$  Matrix



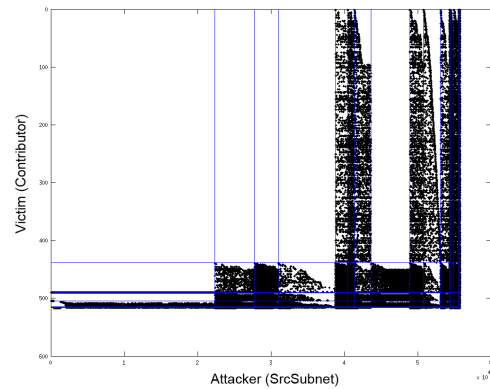
(c) Original  $(c\_id, src\_ip)$  Matrix



(d) Clustered  $(c\_id, src\_ip)$  Matrix

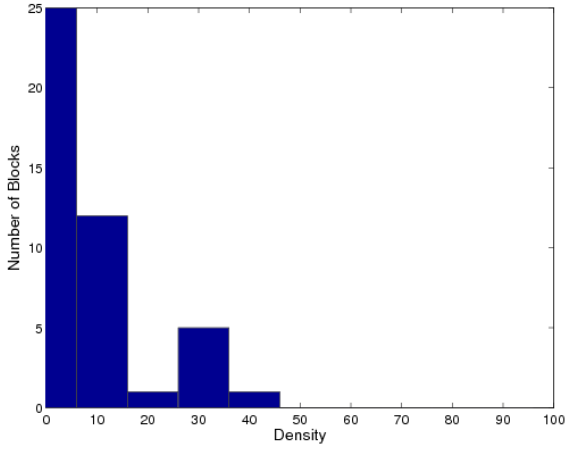


(e) Original  $(c\_id, src\_subnet)$  Matrix

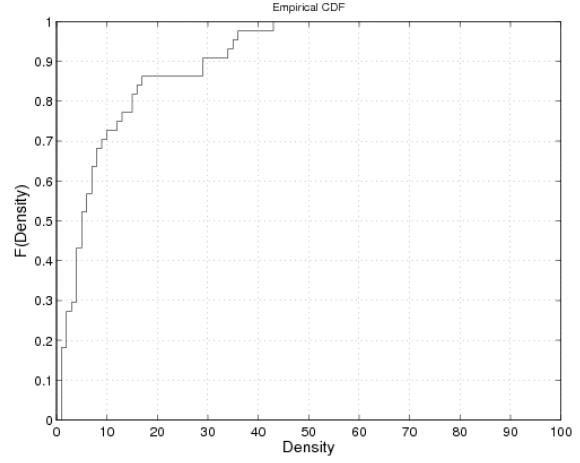


(f) Clustered  $(c\_id, src\_subnet)$  Matrix

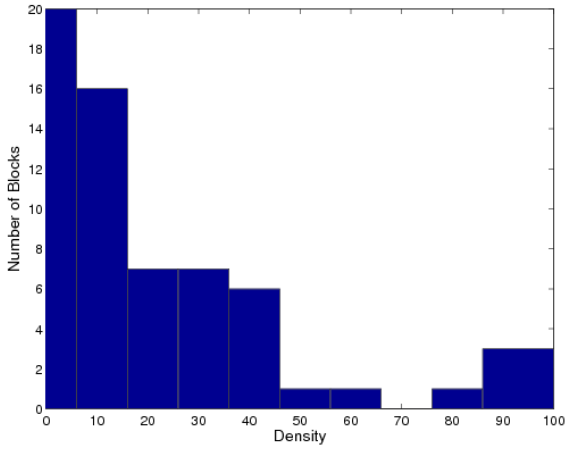
Figure 1: Results of Running The CA Algorithm on Matrices Containing Pairs  $(dst\_ip, src\_ip)$ ,  $(c\_id, src\_ip)$ , and  $(c\_id, src\_subnet)$  of 1-Day Log Entries



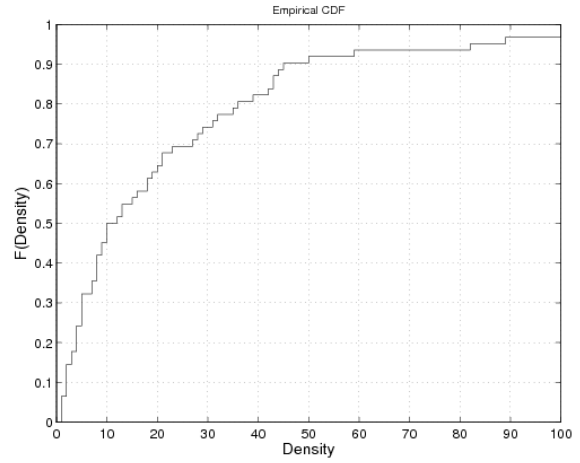
(a) Density Histogram of  $(dst\_ip, src\_ip)$  Clusters



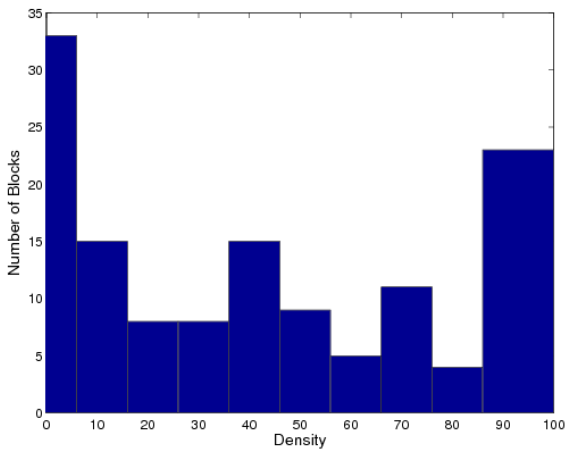
(b) CDF of Density of  $(dst\_ip, src\_ip)$  Clusters



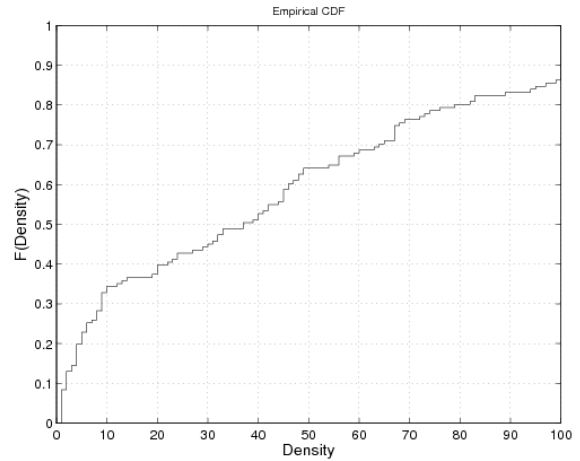
(c) Density Histogram of  $(c\_id, src\_ip)$  Clusters



(d) CDF of Density of  $(c\_id, src\_ip)$  Clusters

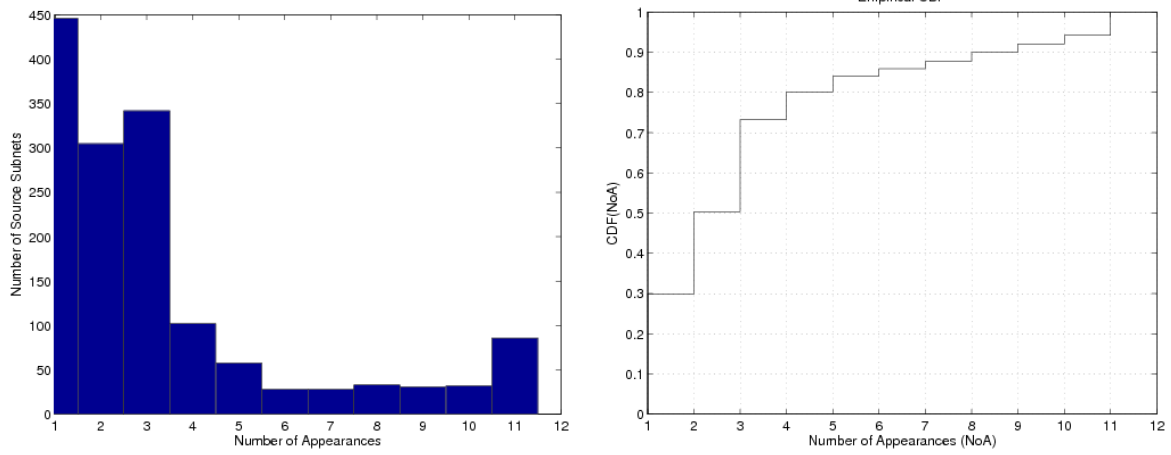


(e) Density Histogram of  $(c\_id, src\_subnet)$  Clusters



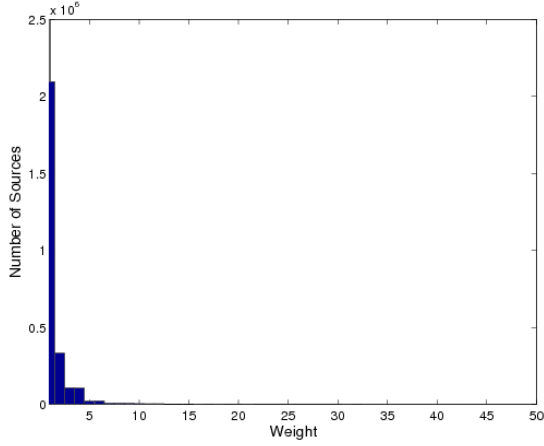
(f) CDF of Density of  $(c\_id, src\_subnet)$  Clusters

Figure 2: Histograms and CDFs of Densities of Clusters of Clustered Matrices Containing Pairs  $(dst\_ip, src\_ip)$ ,  $(c\_id, src\_ip)$ , and  $(c\_id, src\_subnet)$  of 1-Day Log Entries

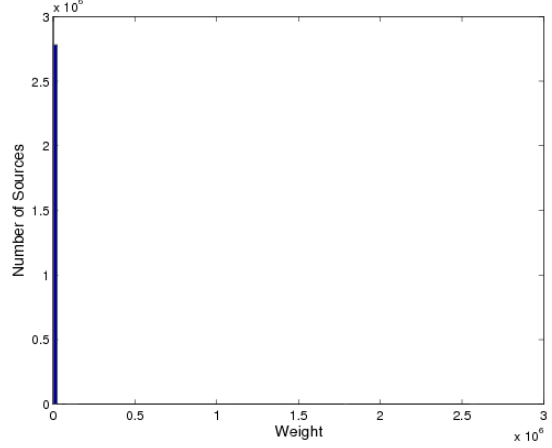


(a) Histogram of Number of Appearances of Botnet Candidates (Source Subnets) over 12 Days (b) CDF of Number of Appearances of Botnet Candidates (Source Subnets) over 12 Days

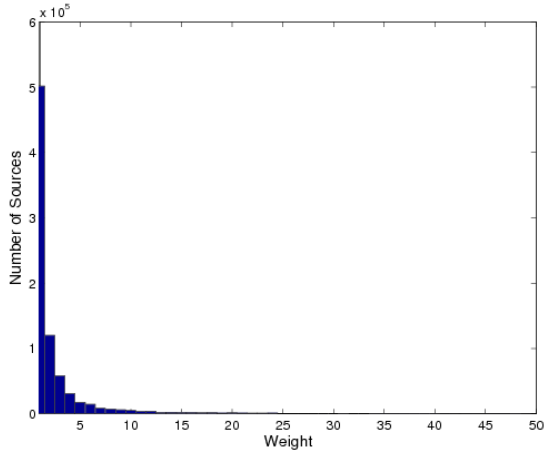
Figure 3: Histogram and CDF of Number of Appearances of Botnet Candidates (Source Subnets) over 12 Days.



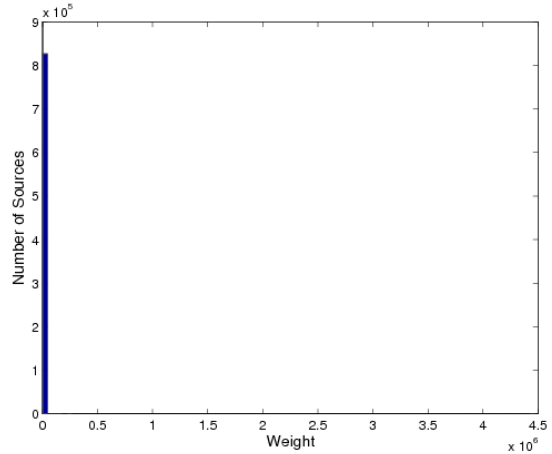
(a) Histogram of Weights of  $(dst\_ip, src\_ip)$  Pairs with Weights up to 50



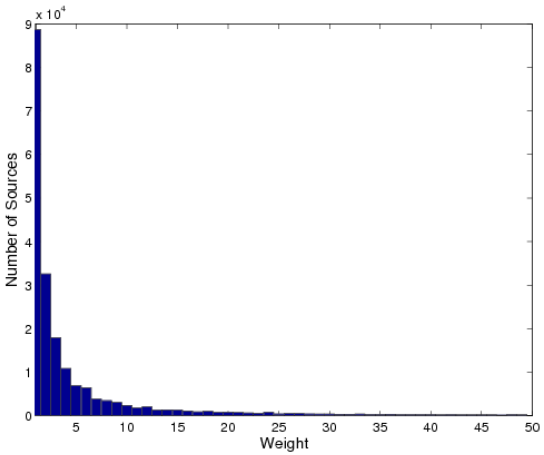
(b) Histogram of Weights of  $(dst\_ip, src\_ip)$  Pairs



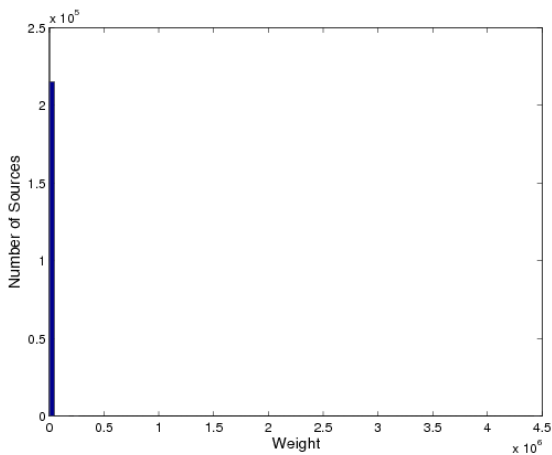
(c) Histogram of Weights of  $(c\_id, src\_ip)$  Pairs with Weights up to 50



(d) Histogram of Weights of  $(c\_id, src\_ip)$  Pairs

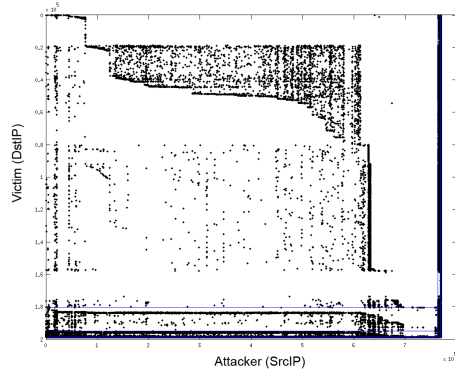


(e) Histogram of Weights of  $(c\_id, src\_subnet)$  Pairs with Weights up to 50

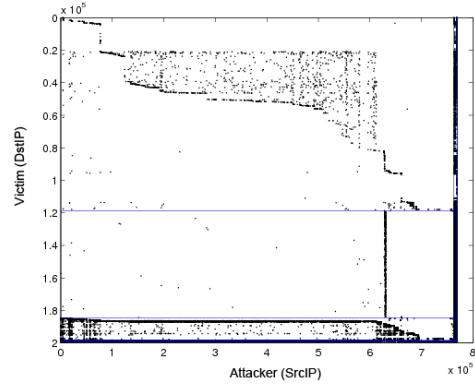


(f) Histogram of Weights of  $(c\_id, src\_subnet)$  Pairs

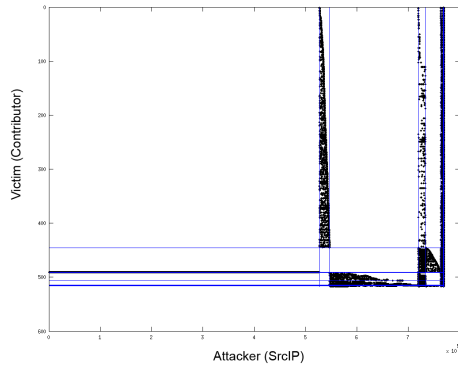
Figure 4: Histograms of Weights of Pairs  $(dst\_ip, src\_ip)$ ,  $(c\_id, src\_ip)$ , and  $(c\_id, src\_subnet)$  of 1-Day Log Entries



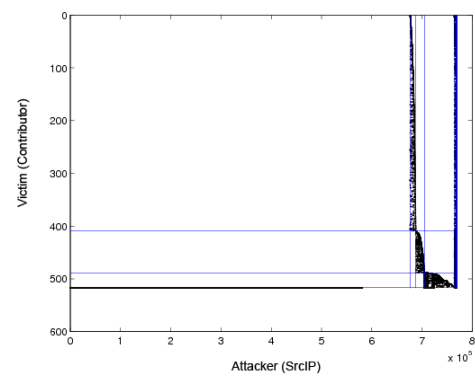
(a) Clustered  $(dst\_ip, src\_ip)$  Matrix of All Pairs



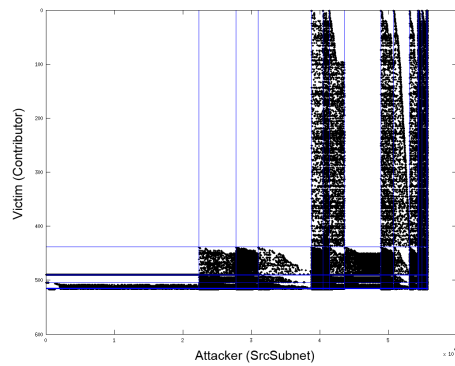
(b) Clustered  $(dst\_ip, src\_ip)$  Matrix of Pairs Appearing at Least Twice



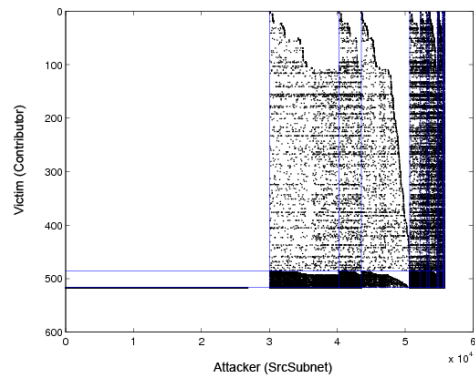
(c) Clustered  $(c\_id, src\_ip)$  Matrix of All Pairs



(d) Clustered  $(c\_id, src\_ip)$  Matrix of Pairs Appearing at Least Twice



(e) Clustered  $(c\_id, src\_subnet)$  Matrix of All Pairs



(f) Clustered  $(c\_id, src\_subnet)$  Matrix of Pairs Appearing at Least Twice

Figure 5: Results of Running The CA Algorithm on Matrices Containing All Pairs and Only Pairs Appearing at Least Twice of 1-Day Log Entries